IN THE SPECIFICATION:

Please amend the Specification as follows:

Paragraph starting at page 38, line 4:

The present invention includes a method and structure for accelerating searches within an address table 21, such as a layer 2 table. Referring to Figure 39, a more detailed view of an accelerated lookup configuration is disclosed with respect to address table 21 and search engine 210. In one example, address table 21 might be a single 8K sorted table that is searched by a single search engine 210. In the accelerated example shown in Figure 39 ~~38~~, this single address table 21 is split into two half-sized tables 211 and 212, with each half-sized table having 4K entries. The table can be split by having address table 211 contain all of the even addressed entries from original address table 21, and table 212 containing all of the odd addressed entries from original address table 21. By splitting the original address table 21 into two separate tables based upon the last bit of the table address, each of tables 211 and 212 remain in sorted order, and contain entries from the entire address range of the original table 21. Search engine 210 can then be divided into two separate search engines, first search engine 213 and second search engine 214, as shown in Figure 39 ~~38~~, which are configured to perform simultaneous address lookups for two data packets. In SOC 10, since each EPIC module 20 and/or GPIC module 30 has a plurality of ports, packets are queued for lookup. Concurrent and/or simultaneous lookups are possible, as the search algorithm which is implemented in SOC 10 does not differentiate between even and odd addressed entries until the very

last search cycle. This optimization, therefore, enables a significant amount of the searching for two separate packet addresses to be performed simultaneously, in parallel, thereby nearly doubling throughput, even though the actual time required to complete each individual lookup does not change.

Paragraph starting at page 60, line 29:

• If there is a partial match and a full match, the counters are updated only for the full ~~march~~ match according to the rules table. If there is only a partial match, then the counters are updated according to action in the filter mask. If all of the filters have a full match in the rules table and the action is to increment the same counter, then the counter is incremented only once. If all of the filters have a partial match and the action is to increment the same counter, then the counter is incremented only once.

Paragraph starting at page 63, line 24:

In using multi-field classifiers, SOC 10 uses FFP mechanism 141 to implement the multi-field (MF) classifications, which are accomplished at the network boundaries. MF classifier capability is implemented using the FFP 141 engine, wherein the filter mask and the corresponding rules table are ~~programed~~ programmed as per the corresponding Differentiated services related policies to assign a new code point or the change the codepoint of the packet. The same rules entry can be used to change 802.1p priority of the packet, depending on the particular policy.

Paragraph starting at page 67, line 14:

Figure 47 shows a detailed flowchart of the logic contained within step 42-4 of Figure 42. At step 47-1 the logic gets the PortBitmap and conducts a logical "and" operation with this value and the forwarding port register, while also "anding" this value with the active port register corresponding to the COS queue selected, after going through the COS mapping using the COS mapping using the COS Select Register. This value is also "anded" with the HOL Register value, which corresponds to Active Port Register 8, to get the PortBitmap at this step. The logic also looks at the M bits of the port based VLAN at this step. Upon completion of the actions of step 47-1, the logic continues to step 47-2, where the logic checks to see if the ingress port is mirrored, that is if the M bit is 0, or if the stack link and the M bit is set. If so, then the packet is sent to the according mirrored port at step 47-3, while if not, then the logic continues to step 47-4 without taking any mirror port action. At step 47-4 the logic checks to ~~se~~ see if the mirroring is based upon filter logic. If so, then the packet is sent to the appropriate mirrored port at step 47-5, while if not, then the logic continues to step 47-6 without taking any mirrored port action. At step 47-6 the logic checks to see if the egress port is mirrored by looking at the egress mirroring register. If the egress port is mirrored, then the packet is sent to the mirrored port before continuing to step 47-8. If the packet is not mirrored, then the logic simply continues directly to step 47-8 without taking any mirror port action. Step 47-8 continues with the mirror port logic, and therefore, will not be discussed in detail.

Nonetheless, at this stage of the logic the DSCP has been accordingly modified such that the packet flow can be appropriately shaped and/or metered upon egress.

Paragraph starting at page 68, line 6:

A5

In another embodiment, the packet flow control logic is folded into the filtering logic, and therefore, the packet flow control logic operates in conjunction with the filtering. In this embodiment, the filtering/flow logic operates in three stages: first, the logic takes actions that are independent of the packet profile, that is the actions do not depend on the classification of the packet as in-profile or out-profile; second, the filtering/flow logic picks up the ~~meterid~~ meter id, which is a 6 bit number associated with the packet that is stored in the rules table, and takes any appropriate in-profile actions that are set; and third, the filtering/flow logic takes any appropriate out-profile actions. Beginning with the profile independent actions, upon application of each individual filter mask, which is generally undertaken in ascending numerical order, a determination is made by the filtering/flow logic as to whether there is a matching rule, as shown in Figure 32. This determination essentially determines whether or not there is a full match, as previously defined, which is shown in Figure 32 as step 32-1. If it is determined that there is a full match for the particular mask at step 32-1, then the first action bit is checked at step 32-2. If a full match is not determined at step 32-1, then the logic determines whether or not there is a partial match for the mask at step 32-3. If a partial match is found at step 32-3, then the logic continues through the partial match method,

which is illustrated in Figure 33 and will be further discussed below. Returning to step 32-2, if it is determined that bit 1 of the action bits is set, then the class of service is selected from this rule entry at step 32-4. If bit 1 of the action bits is not set, then the logic continues to step 32-5 and checks to see if bit 0 of the action fields is set. If bit 0 is set, then the class of service for this entry is obtained from the rule entry, the packet is modified for priority tagged field, and the regenerate CRC bit is set at step 32-5. If bit 0 is not set, then the logic continues to step 32-6, which generally represents the beginning of the flow control logic, and the end of the profile-independent actions, as action bit 0 and action bit 1 are independent actions from flow control. Put simply, the profile independent actions are taken at steps 32-1 through 32-5, and beginning at step 32-6, the profile dependent actions begin. At step 32-6, the meter id for the particular mask is obtained from the rules table. At step 32-7, the logic determines if the meter id obtained from the rules table is 0. If the meter id is 0, then the packet is automatically judged to be in-profile, and the appropriate in-profile actions are immediately taken at step 32-8. If the meter id is not 0, then the logic indexes into the meter table with the meter id at step 32-9 to determine the profile status of the packet. At step 32-10, upon indexing into the meter table, the logic determines if the packet is in fact in-profile, and if so, then the in-profile actions of step 32-8 are taken. If the packet is not determined to be in-profile, then by default the packet is determined to be out-profile at step 32-11. Therefore, upon the determination that the packet is out-profile, the appropriate out-profile actions are taken at step 32-12.

Paragraph starting at page 85, line 22:

Figure 25 illustrates a flowchart for how ingress 14 of an SOC ~~20~~ 10 would handle an IP multicast packet coming in to a port thereupon. In step 25-1, the packet is examined to determine whether or not it is an IP multicast packet without any option fields. If there are option fields, the packet is sent to CPU 52 for further handling. In step 25-2, the IP checksum is validated. In step 25-3, the destination IP address is examined to see if it is a class D address. A class D address is one where the first three most significant bits are all set to 1. If the destination IP address is not a class D address, then the packet is dropped. If so, the IP multicast table is searched at step 25-4 with the key as the source IP address + the destination IP address. If the entry is not found, then the packet is sent to CPU 52. If a match is found at step 25-5, the TTL (time-to-live) is checked against the TTL threshold value in the IP multicast entry at step 25-6. If the TTL value is less than the threshold value, the packet is dropped. If the TTL value is not below the threshold, then the source port is compared to the source port in the entry at step 25-7. If there is not a match, then the packet is dropped. If there is a match, the packet is appropriately sent over C channel 81 of CPS channel 80 at step 25-8, with appropriate P channel messages locked therewith. The packet is sent with the L2 port bitmap and L2 untagged bitmap obtained as a result of the L2 search, and the L3 port bitmap as a result of the IP multicast search. Additionally, the IP multicast bit is set in the P channel message, indicating that the packet is an IP multicast packet and that the egress, upon receipt of the packet, must modify the IP header appropriately. From CPS

channel 80, therefore, the packet is sent to the appropriate buffer pool until it is obtained by the appropriate egress port.

Paragraph starting at page 102, line 21:

Referring to Figure 29, address resolution for a packet coming in to IPIC 90 from high performance interface ~~271~~ 261 is as follows:

Paragraph starting at page 114, line 6:

These situations, which are discussed with respect to ~~Figured~~ Figures 48 – 51, each result in one of three outcomes. Outcome number one represents a re-transmission state, wherein the watchdog timer determines that an ACK has not been received within a predetermined period of time, and therefore, resends the REQ. The second outcome represents normal operation mode outcome wherein the REQ is properly acknowledged. However, one flow control path discussed above leads to outcome number 2 despite a data corruption. This is the unlikely result of a double corruption, wherein the original REQ ID is corrupted followed by a corruption of the ACK that essentially operates to "undo" the previous corruption, as discussed with respect to step 51 – 8, 9, and 10. The third outcome is by far the most unlikely outcome, as this outcome requires a 2+ bit error that does not result in a detection, as the corruption happens to match a previously transmitted REQ that has not yet been acknowledged with a valid ACK. The three outcome situations provided by the present invention address each situation that may

generally be encountered by high speed interface control logic operating to transmit data between two stations. In particular, the flow control logic of the present invention provides a solution to all generally occurring flow control characteristics that would be encountered if the two stations represent a pair of network switches that are interconnected through a high speed interconnect, either in a stacking configuration or other known configuration.